Distributed Function Calculation and Consensus using Linear Iterative Strategies

Shreyas Sundaram and Christoforos N. Hadjicostis

***Abstract*— Given an arbitrary network of interconnected nodes, we develop and analyze a distributed strategy that enables a subset of the nodes to calculate any given function of the node values. Our scheme utilizes a linear iteration where, at each time-step, each node updates its value to be a weighted average of its own previous value and those of its neighbors. We show that this approach can be viewed as a linear dynamical system, with dynamics that are given by the weight matrix of the linear iteration, and with outputs for each node that are captured by the set of values that are available to that node at each time-step. In networks with time-invariant topologies, we use observability theory to show that after running the linear iteration for a finite number of time-steps with almost any choice of weight matrix, each node obtains enough information to calculate any arbitrary function of the initial node values. The problem of distributed consensus via linear iterations, where all nodes in the network calculate the same function, is treated as a special case of our approach. In particular, our scheme allows nodes in networks with time-invariant topologies to reach consensus on any arbitrary function of the initial node values in a finite number of steps for almost any choice of weight matrix.**

***Index Terms*— Distributed function calculation, distributed consensus, observability theory, structural observability, networked control, multi-agent systems**

I. Introduction

In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of certain parameters. For example, sink nodes in sensor networks may be tasked with calculating the average measurement value of all the sensors [1], [2]. Another example is the case of multi-agent systems, where all agents communicate with each other to coordinate their speed and direction [3], [4]. When all nodes calculate the same function of the initial values in the system, they are said to reach consensus. Such problems have received extensive attention in the computer science literature over the past few decades [5], leading to the development of various protocols. The topic of distributed consensus has also attracted the attention of the control community due to its applicability to tasks such as coordination of multi-agent systems [3]. In these cases, the approach to consensus is to use a linear iteration, where each node repeatedly updates its value as a weighted linear combination of its own value and those of its neighbors [6], [7], [8], [9], [10]. These works have revealed that if the network topology satisfies certain conditions, the weights for the linear iteration can be chosen so that all of the nodes asymptotically converge to the same value (even when the topology is time-varying). One of the benefits of using linear iteration-based consensus schemes is that, at each time-step, each node only has to transmit a single value to each of its neighbors. However, almost all of the linear iteration schemes currently present in the literature only produce asymptotic convergence (i.e., exact consensus is not reached in a finite number of steps).

In this paper, we study the general problem of calculating functions in networks via a linear iteration, and treat the topic of distributed consensus as a special case of this problem. We approach the problem from the perspective of observability theory, and show that the linear iteration-based scheme can be modeled as a dynamical system. For networks with time-invariant topologies, we utilize results that have been developed for the analysis of structured system observability to show that for almost any choice of weight matrix, each node can calculate any desired function of the initial values after observing the evolution of its own value and those of its neighbors over a *finite* number of time-steps (specifically, upper bounded by the size of the network). The topic of finite-time consensus via linear iterations has received only limited attention in the literature. Specifically, it was briefly discussed by Kingston and Beard in [11], but the method described in that paper requires the network to be fully-connected (i.e., every node needs to be directly connected to every other node) for at least one time-step, which is a very strict condition. In contrast, our method applies to networks with arbitrary (but time-invariant) topologies. Finite-time consensus was also studied for continuous-time systems in [12]. The approach in that paper was to have the nodes evolve according to the gradient of a suitably defined function of their values. The resulting protocol, which is nonlinear, does not directly translate to discrete-time systems, and the author of [12] only considered a restricted class of possible consensus values. In contrast, our method achieves finite-time consensus in discrete-time systems by running a simple linear iteration, and allows the consensus value to be any function of the node values.

The rest of the paper is organized as follows. In Section II, we provide a more detailed background on linear iteration-based consensus schemes, and introduce the key result in

This material is based upon work supported in part by the National Science Foundation under NSF Career Award 0092696 and NSF ITR Award 0426831. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF. Some of the results in this paper were presented in preliminary form at the 2007 International Conference on Information Processing in Sensor Networks (IPSN).

The authors are with the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA. E-mail: {ssundarm, chadjic}@uiuc.edu.

Corresponding author: Christoforos N. Hadjicostis. Address for correspondence: 357 CSL, 1308 West Main St., Urbana, IL, 61801-2307, USA. Phone: (217)265-8259. Fax: (217)244-1653.

our paper. In Section III, we review some concepts from observability theory that will be useful in solving the problem of function calculation in networks. We use these concepts in Section IV to design weight matrices that allow the desired functions to be calculated at certain nodes. In Section V, we discuss how to implement our scheme in a decentralized manner. We present an example in Section VI, and we finish with our conclusions and future research directions in Section VII.

In our development, we use $\mathbf{e}_i$ to denote the column vector with a 1 in its i–th position and zeros elsewhere. The symbol $\mathbf{1}$ represents the column vector (of appropriate size) that has all entries equal to one. We will say that an eigenvalue of a matrix W is *simple* to indicate that it is *algebraically simple* (i.e., it appears only once in the spectrum of W) [13]. The notation A' indicates the transpose of matrix A. We will denote the rank of matrix A by $\rho(A)$, and we will denote the cardinality of a set S by $|S|$.

II. Background and Contributions

The interaction constraints in distributed systems and networks can be conveniently modeled via a directed graph $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the set of nodes in the system and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of directed edges (i.e., directed edge $(x_j, x_i) \in \mathcal{E}$ if node x_i can receive information from node x_j). Note that undirected graphs can be readily handled by treating each undirected edge as two directed edges. All nodes that can transmit information to node x_i are said to be neighbors of node i, and are represented by the set $\mathcal{N}_i$. The number of neighbors of node i is called the in-degree of node i, and is denoted as $\deg_i$. Similarly, the number of nodes that have node i as a neighbor is called the out-degree of node i, and is denoted as out-$\deg_i$.

Suppose that each node i has some initial value, given by $x_i[0]$. At each time-step k, all nodes update and/or exchange their values based on some strategy that adheres to the constraints imposed by the network topology (which is assumed to be time-invariant). The scheme that we study in this paper makes use of linear iterations; specifically, at each time-step, each node updates its value as

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{j \in \mathcal{N}_i} w_{ij}x_j[k] \ ,$$

where the w_{ij}'s are a set of weights. In other words, each node updates its value to be a linear combination of its own value and the values of its neighbors. For ease of analysis, the values of all nodes at time-step k can be aggregated into the value vector $x[k] = \begin{bmatrix} x_1[k] & x_2[k] & \cdots & x_N[k] \end{bmatrix}'$, and the update strategy for the entire system can be represented as

$$x[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{W} x[k] \qquad (1)$$

for $k = 0, 1, \ldots$, where $w_{ij} = 0$ if $j \notin \mathcal{N}_i$ (i.e., if $(x_j, x_i) \notin \mathcal{E}$).

Definition 1 (Calculable Function): Let $f : \mathbb{R}^N \mapsto \mathbb{R}^m$ be a function of the initial values of the nodes (note that $f(\cdot)$ will be a vector-valued function if $m \geq 2$). We say $f(x_1[0], x_2[0], \ldots, x_N[0])$ is *calculable by node* i if it can be calculated by node i after running the linear iteration (1) for a sufficiently large number of time-steps. We call $f(x_1[0], x_2[0], \ldots, x_N[0])$ a *linear functional* if it is of the form $Qx[0]$ for some $m \times N$ matrix Q. □

Definition 2 (Distributed Consensus and Averaging): The system is said to achieve distributed consensus if all nodes in the system calculate the value $f(x_1[0], x_2[0], \ldots, x_N[0])$ after running the linear iteration for a sufficiently large number of time-steps. When

$$f(x_1[0], x_2[0], \ldots, x_N[0]) = \frac{1}{N}\mathbf{1}'x[0] \ ,$$

the system is said to perform *distributed averaging* (i.e., the consensus value is the average of all initial node values). □

Definition 3 (Asymptotic Consensus): The system is said to reach asymptotic consensus if $\lim_{k\to\infty} x_i[k] = f(x_1[0], x_2[0], \ldots, x_N[0])$ for each i, where $f(x_1[0], x_2[0], \ldots, x_N[0]) \in \mathbb{R}$. □

When $f(x_1[0], x_2[0], \ldots, x_N[0]) = \mathbf{c}'x[0]$ for some vector $\mathbf{c}'$, the following result by Xiao and Boyd from [8] characterizes the conditions under which the iteration (1) achieves asymptotic consensus.

Theorem 1 ([8]): The iteration given by (1) reaches asymptotic consensus on the linear functional $\mathbf{c}'x[0]$ (under the technical condition that $\mathbf{c}$ is normalized so that $\mathbf{c}'\mathbf{1} = 1$) if and only if the weight matrix W satisfies the following conditions:

1) W has a simple eigenvalue at 1, with left eigenvector $\mathbf{c}'$ and right eigenvector $\mathbf{1}$.
2) All other eigenvalues of W have magnitude strictly less than 1.

□

Furthermore, Xiao and Boyd showed that the rate of convergence is determined by the eigenvalue with second-largest magnitude. Consequently, to maximize the rate of convergence, they formulated a convex optimization problem to obtain the weight matrix satisfying the above conditions while minimizing the magnitude of the second largest eigenvalue.

As mentioned in the previous section, a salient feature of the analysis in [8], and of other linear iterations studied in the literature (e.g., see [3], [4] and the references therein), is that the focus is on asymptotic convergence. Clearly, reaching consensus on $\mathbf{c}'x[0]$ requires that the linear functional $\mathbf{c}'x[0]$ be calculable at all nodes, and in the case of asymptotic consensus, the nodes take an infinite number of time-steps in order to calculate this functional. While one might have to settle for asymptotic consensus when the network topology is time-varying, we will show in the sequel that in time-invariant networks, nodes can calculate any arbitrary function in a distributed fashion after running the linear iteration (1) for a *finite* number of time-steps. Specifically, we will demonstrate the following key result.

Theorem 2: Let $\mathcal{G}$ denote the (fixed) graph of the network, and

$$S_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup x_i \ .$$

Then, for almost any choice of weight matrix W, node i can obtain the value $x_j[0]$, $x_j \in S_i$, after running the linear iteration (1) for $L_i + 1$ time-steps, for some $0 \leq L_i < |S_i| - \deg_i$, and can therefore calculate any arbitrary function of the values $\{x_j[0] \mid x_j \in S_i\}$. □

We will develop the proof of this theorem over the next few sections. We will also show that the above theorem immediately allows the nodes to obtain (finite-time) distributed consensus on a much larger class of functions than permitted by Theorem 1 (e.g., one does not have to restrict the consensus value to be a linear functional). Moreover, since we focus on the problem of function calculation rather than the more specific problem of distributed consensus, our method can be applied to a wide range of problems, including those where only a subset of the nodes have to calculate a given function, or where different nodes need to calculate different functions of the initial values.

III. Distributed Computation of Linear Functionals

We start by asking the question: after running the linear iteration (1) with a given weight matrix[1] W for $L_i + 1$ time-steps (for some L_i), what is the set of all linear functionals that are calculable by node i? The following lemma provides an answer to this question.

Lemma 1: Define E_i to be the $(\deg_i + 1) \times N$ matrix with a single 1 in each row, denoting the positions of the vector $x[k]$ that are available to node i (i.e., these positions correspond to nodes that are neighbors of node i, along with node i itself). Then, node i can calculate the linear functional $Qx[0]$ after running the linear iteration (1) for $L_i + 1$ time-steps if and only if the row space of Q is contained in the row space of the matrix

$$\mathcal{O}_{i,L_i} \equiv \begin{bmatrix} E_i \\ E_i W \\ E_i W^2 \\ \vdots \\ E_i W^{L_i} \end{bmatrix} \quad . \tag{2}$$

□

Proof: Consider the linear iteration given by (1). At each time-step, node i has access to its own value and those of its neighbors. From the perspective of node i, the linear iteration can then be viewed as the dynamical system

$$\begin{aligned} x[k+1] &= Wx[k] \\ y_i[k] &= E_i x[k] \ , \end{aligned} \tag{3}$$

where $y_i[k]$ denotes the outputs (node values) that are seen by node i during the k–th time-step. Since $x[k] = W^k x[0]$, we have $y_i[k] = E_i W^k x[0]$, and the set of all values seen by node i over $L_i + 1$ time-steps is given by

$$\begin{bmatrix} y_i[0] \\ y_i[1] \\ y_i[2] \\ \vdots \\ y_i[L_i] \end{bmatrix} = \underbrace{\begin{bmatrix} E_i \\ E_i W \\ E_i W^2 \\ \vdots \\ E_i W^{L_i} \end{bmatrix}}_{\mathcal{O}_{i,L_i}} x[0] \ . \tag{4}$$

The row space of $\mathcal{O}_{i,L_i}$ characterizes the set of all calculable linear functionals for node i up to time-step L_i. Suppose that the row space of matrix Q is not contained in the row space of $\mathcal{O}_{i,L_i}$. This means that

$$\rho\left(\begin{bmatrix} \mathcal{O}_{i,L_i} \\ Q \end{bmatrix}\right) > \rho\left(\mathcal{O}_{i,L_i}\right) \ ,$$

and so there exists a nonzero vector $\mathbf{v}$ such that $\mathcal{O}_{i,L_i}\mathbf{v} = 0$, but $Q\mathbf{v} \neq 0$. If $x[0] = \mathbf{v}$, the values seen by node i during the first $L_i + 1$ time-steps of the linear iteration are all zeros, and so node i cannot calculate $Q\mathbf{v}$ from the outputs of the system (i.e., it cannot distinguish the initial value vector $x[0] = \mathbf{v}$ from the case where all initial values are zero).

On the other hand, if the row space of Q is contained in the row space of $\mathcal{O}_{i,L_i}$, one can find a matrix Γ_i such that

$$\Gamma_i \mathcal{O}_{i,L_i} = Q \ . \tag{5}$$

Thus, after running the linear iteration (1) for $L_i + 1$ time-steps, node i can immediately calculate $Qx[0]$ as a linear combination of the outputs of the system over those time steps, i.e.,

$$\Gamma_i \begin{bmatrix} y_i[0] \\ y_i[1] \\ \vdots \\ y_i[L_i] \end{bmatrix} = \Gamma_i \mathcal{O}_{i,L_i} x[0] = Qx[0] \ . \tag{6}$$

This concludes the proof of the lemma. ■

Remark 1: The above lemma showed (via equation (6)) that node i can obtain the functional $Qx[0]$ after running the linear iteration for $L_i + 1$ time-steps (for some L_i, and assuming that Q is in the row space of $\mathcal{O}_{i,L_i}$). Note that node i does not necessarily have to store the entire set of values $y_i[0], y_i[1], \ldots, y_i[L_i]$ in order to calculate the quantity $Qx[0]$ via (6). Instead, if we partition Γ_i as $\Gamma_i = \begin{bmatrix} \Gamma_{i,0} & \Gamma_{i,1} & \cdots & \Gamma_{i,L_i} \end{bmatrix}$ and allow node i to have m extra registers (where m is the number of rows in Q), we can utilize the following recursive strategy: we initialize the values of the m registers with $\bar{x}_i[0] = \Gamma_{i,0} y_i[0]$ and update them at each time-step k as $\bar{x}_i[k] = \bar{x}_i[k-1] + \Gamma_{i,k} y_i[k]$. After L_i iterations, $\bar{x}_i[L_i]$ will hold the value of $Qx[0]$. □

The matrix $\mathcal{O}_{i,L_i}$ in (2) is called the *observability matrix* for the pair (W, E_i), and it plays an important role in linear system theory [14]. Note that the rank of $\mathcal{O}_{i,L_i}$ is a nondecreasing function of L_i, and bounded above by N. Suppose ν_i is the first integer for which $\rho(\mathcal{O}_{i,\nu_i}) = \rho(\mathcal{O}_{i,\nu_i - 1})$. This implies

[1] We will discuss ways to choose appropriate weight matrices in the next section.

that there exists a matrix $\mathcal{K}_i$ such that

$$E_i W^{\nu_i} = \mathcal{K}_i \begin{bmatrix} E_i \\ E_i W \\ \vdots \\ E_i W^{\nu_i - 1} \end{bmatrix} .$$

In other words, the matrix $E_i W^{\nu_i}$ can be written as a linear combination of the rows in $\mathcal{O}_{i,\nu_i - 1}$. In turn, this implies that

$$E_i W^{\nu_i + 1} = E_i W^{\nu_i} W = \mathcal{K}_i \mathcal{O}_{i,\nu_i - 1} W = \mathcal{K}_i \begin{bmatrix} E_i W \\ E_i W^2 \\ \vdots \\ E_i W^{\nu_i} \end{bmatrix} ,$$

and so the matrix $E_i W^{\nu_i + 1}$ can be written as a linear combination of rows in $\mathcal{O}_{i,\nu_i}$. Continuing in this way, we see that

$$\rho(\mathcal{O}_{i,0}) < \rho(\mathcal{O}_{i,1}) < \cdots < \rho(\mathcal{O}_{i,\nu_i - 1}) = \rho(\mathcal{O}_{i,\nu_i}) = \rho(\mathcal{O}_{i,\nu_i + 1}) = \cdots ,$$

i.e., the rank of $\mathcal{O}_{i,L_i}$ monotonically increases with L_i until $L_i = \nu_i - 1$, at which point it stops increasing. In the linear systems literature, the integer ν_i is called the *observability index* of the pair (W, E_i), and can be upper bounded as $\nu_i \leq \min(D, N - \rho(E_i) + 1)$, where D is the degree of the minimal polynomial[2] of W [14]. Since E_i has rank $\deg_i + 1$ in the linear iteration model (3), the above bound becomes

$$\nu_i \leq \min(D, N - \deg_i) . \tag{7}$$

The fact that the rank of the observability matrix stops increasing after time-step $\nu_i - 1$ means that the outputs of the system $y_i[0], y_i[1], \ldots, y_i[\nu_i - 1]$ contain the maximum amount of information that node i can possibly obtain about the initial values, and future outputs of the system do not provide any extra information. This immediately produces the following lemma.

Lemma 2: If it is possible for node i to obtain the necessary information to calculate the linear functional $Qx[0]$ via the linear iteration (1), it will require at most $N - \deg_i$ time-steps. □

Remark 2: Note that the above lemma only provides an upper bound on the number of time-steps required by any node to calculate any desired functional. A lower bound on the number of time-steps required by node i can be obtained as follows. Let $\mathcal{F}_i$ denote the set of all nodes whose values are required by node i in order to calculate its function (if node i's function depends on all initial values, the set $\mathcal{F}_i$ will contain all nodes in the graph). Pick a node $x_j \in \mathcal{F}_i$ that is farthest away from node i in the network, and let the distance[3] between node j and node i be denoted by ϵ_i. Since it takes one time-step for a value to propagate along an edge, it will take at least ϵ_i time-steps before node i is able to obtain node j's value, and thus a lower bound on the number of time-steps required for function calculation by node i is ϵ_i. Note that if $\mathcal{F}_i$ contains all nodes in the network, ϵ_i is simply the eccentricity of node i [15]. In particular, if $\epsilon_i = N - \deg_i$, then node i will take exactly $N - \deg_i$ time-steps to calculate its desired function (since the lower bound and upper bound coincide in this case). □

If the objective in the system is for some subset of the nodes to calculate the linear functional $Qx[0]$, one has to choose the weight matrix W so that Q is in the row space of the observability matrices for all those nodes. More generally, suppose we require node i to calculate the function $g(x_{t_1}[0], x_{t_2}[0], \ldots, x_{t_S}[0])$, where $\{x_{t_1}, x_{t_2}, \ldots x_{t_S}\}$ is some subset of the nodes in the system. If W can be designed so that the matrix $Q = \begin{bmatrix} \mathbf{e}_{t_1} & \mathbf{e}_{t_2} & \cdots & \mathbf{e}_{t_S} \end{bmatrix}'$ is in the row space of $\mathcal{O}_{i,\nu_i - 1}$, node i can recover the initial values $x_{t_1}[0], x_{t_2}[0], \ldots, x_{t_S}[0]$ from $y_i[0], y_i[1], \ldots, y_i[\nu_i - 1]$, and then calculate $g(x_{t_1}[0], x_{t_2}[0], \ldots, x_{t_S}[0])$. If $\rho(\mathcal{O}_{i,\nu_i - 1}) = N$ (i.e., the identity matrix is in the row space of the observability matrix), the pair (W, E_i) is said to be *observable*. In this case, node i can uniquely determine the entire initial value vector $x[0]$ from the outputs of the system and calculate any function of those values. Based on this discussion, we see that we can recast the function calculation problem as a weight matrix design problem, where the objective is to make the row space of the observability matrix for each node contain some appropriate matrix Q. We describe how to do this in the next section.

IV. Designing the Weight Matrix

We will start this section by considering the problem of reaching consensus on a linear functional $\mathbf{c}'x[0]$, for some vector $\mathbf{c}'$. We will then broaden our focus in the second half of the section, and show how to design the weight matrix to maximize the set of functions that can be calculated by each node after running the linear iteration for a sufficiently large (but finite) number of time-steps.

A. Reaching Consensus on a Linear Functional

In the last section, we saw that if the vector $\mathbf{c}'$ appears in the row space of the observability matrix for every node, the system can reach consensus on $\mathbf{c}'x[0]$ after running the linear iteration for a finite number of time-steps. We now develop a set of weight matrices that ensures that this is the case. In the process, we will show that this set contains all weight matrices that satisfy the conditions in Theorem 1. To proceed with the development, we will require the notion of *observable eigenvalues* of a pair (W, E_i) [16].

Definition 4 (Observable Eigenvalue): Suppose that the distinct eigenvalues of W are given by $\lambda_1, \lambda_2, \ldots, \lambda_r$, where $1 \leq r \leq N$. Eigenvalue λ_j is said to be an observable eigenvalue of the pair (W, E_i) if

$$\rho\left(\begin{bmatrix} W - \lambda_j I \\ E_i \end{bmatrix} \right) = N , \tag{8}$$

and it is called an unobservable eigenvalue otherwise. □

[2]Note that the degree D of the minimal polynomial of W is always smaller than or equal to N [14].

[3]Recall that the distance between node i and node j in a graph is the smallest number of edges in a path from node j to node i in the graph [15].

With the notion of observable eigenvalues in hand, we now present a theorem that guarantees that a certain class of vectors will be in the row space of the observability matrix.

Theorem 3: The row space of $\mathcal{O}_{i,\nu_i-1}$ contains all left eigenvectors of W corresponding to observable eigenvalues of the pair (W, E_i). □

The proof of the above theorem is given in the Appendix. The theorem provides a strategy for choosing the weight matrix so that a vector $\mathbf{c}'$ will be contained in the row space of the observability matrix for every node. Specifically, the task will be to choose the weight matrix to have an eigenvalue μ with a left eigenvector $\mathbf{c}'$, so that μ is an observable eigenvalue of the pair (W, E_i) for all i. To accomplish this, we first provide a characterization of the observability of an eigenvalue based on its right eigenvector.

Lemma 3: Suppose μ is a simple eigenvalue of W, with right eigenvector $\mathbf{d}$. Then μ is an observable eigenvalue of the pair (W, E_i) if and only if $E_i\mathbf{d}$ has at least one nonzero entry. □

Proof: Suppose μ is a simple eigenvalue of W. Then the matrix $W - \mu I$ has a single eigenvalue at zero, with a null space spanned by the eigenvector $\mathbf{d}$. This implies that the only possible vector in the null space of the matrix $\begin{bmatrix} W-\mu I \\ E_i \end{bmatrix}$ is $\mathbf{d}$. If $E_i\mathbf{d}$ has at least one nonzero entry, the matrix $\begin{bmatrix} W-\mu I \\ E_i \end{bmatrix}$ will have a null space of dimension zero, and therefore μ will be an observable eigenvalue of the pair (W, E_i). On the other hand, if all entries of $E_i\mathbf{d}$ are zero, $\begin{bmatrix} W-\mu I \\ E_i \end{bmatrix}$ will have a null space of dimension one, implying that μ is an unobservable eigenvalue of the pair (W, E_i). ■

In light of the above lemma, suppose we consider the class of weight matrices described in Theorem 1. Such matrices have a simple eigenvalue at $\mu = 1$, with a right eigenvector $\mathbf{d} = \mathbf{1}$ and left eigenvector $\mathbf{c}'$. Since each E_i matrix has a single 1 in each row, the vector $E_i\mathbf{d}$ will be the all ones vector, and so Lemma 3 indicates that the eigenvalue $\mu = 1$ is an observable eigenvalue of the pair (W, E_i) for all $1 \leq i \leq N$. Theorem 3 then indicates that $\mathbf{c}'x[0]$ will be calculable by all nodes. Therefore, if we require the system to reach consensus on $\mathbf{c}'x[0]$, one strategy is to choose W to satisfy the conditions in Theorem 1. For example, if the graph is undirected and if each node knows N (or an upper bound for N), it was shown in [17] that each node i can independently choose its weights as

$$w_{ij} = \begin{cases} \frac{1}{N}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{if } j \notin \mathcal{N}_i \\ 1 - \sum_{l \in \mathcal{N}_i} w_{il}, & \text{if } i = j. \end{cases} \tag{9}$$

The resulting weight matrix is symmetric and satisfies the conditions of Theorem 1 with $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. Many other choices of weights exist that provide asymptotic consensus (e.g., see [8], [3], [7], [10], [11]), and any of these weights can be used in conjunction with our approach to obtain finite-time consensus.

However, we do not necessarily need to restrict attention to matrices that satisfy the conditions in Theorem 1. For example, since Lemma 3 and Theorem 3 do not depend on the magnitude of the eigenvalues, we can consider the weights

$$w_{ij} = \begin{cases} 1, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{if } j \notin \mathcal{N}_i \\ F - \deg_i, & \text{if } i = j \ , \end{cases} \tag{10}$$

where F is any integer. These weights are obtained by multiplying the weights in (9) by N, and then offsetting each of the diagonal weights by a constant amount $F - N$. The corresponding weight matrix has a simple eigenvalue at $\mu = F$, with right eigenvector $\mathbf{d} = \mathbf{1}$ and left eigenvector $\mathbf{c}' = \frac{1}{N}\mathbf{1}'$. This choice of weight matrix also allows all nodes to calculate the functional $\mathbf{c}'x[0]$, and has the appealing feature of being an integer matrix, which could potentially simplify the implementation of the consensus protocol. This is a valid weight matrix in our scheme, but it is not allowed in asymptotic consensus protocols since it will typically have eigenvalues with magnitude larger than 1.

While the analysis and weight matrices described in this section are useful for reaching consensus on a linear functional, it is not clear whether they allow the nodes to calculate more arbitrary functions of the node values, or whether they allow different nodes to calculate different functions. We will now describe how to choose the weights to maximize the set of functions that can be calculated by each node after running the linear iteration for a finite number of time-steps. In the process, we will obtain the proof of Theorem 2.

B. Characterization of All Calculable Functions for Each Node

To determine the set of functions than can be calculated by each node i, we will first examine the maximum amount of information that can be recovered about the initial values after running the linear iteration. This essentially amounts to maximizing the rank of the observability matrix for each node i. To facilitate this investigation, recall the set

$$S_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup x_i$$

(originally defined in Theorem 2), and let $\bar{S}_i$ denote its complement ($\mathcal{G}$ is the graph of the network). Let $x_{S_i}[k]$ denote the vector that contains the values of the nodes in S_i, and let $x_{\bar{S}_i}[k]$ denote the vector of values for the remaining nodes. Without loss of generality, assume that the vector $x[k]$ in (3) has the form $x[k] = \begin{bmatrix} x'_{S_i}[k] & x'_{\bar{S}_i}[k] \end{bmatrix}'$ (the vector can always be arranged in this form by an appropriate permutation of the node indices). Since there is no path from any node in $\bar{S}_i$ to node i (and hence no path from any node in $\bar{S}_i$ to any node in S_i), equation (3) takes the form

$$\begin{bmatrix} x_{S_i}[k+1] \\ x_{\bar{S}_i}[k+1] \end{bmatrix} = \begin{bmatrix} W_{i,S_i} & 0 \\ W_{i,S_i\bar{S}_i} & W_{i,\bar{S}_i} \end{bmatrix} \begin{bmatrix} x_{S_i}[k] \\ x_{\bar{S}_i}[k] \end{bmatrix}$$
$$y_i[k] = \begin{bmatrix} E_{i,S_i} & 0 \end{bmatrix} \begin{bmatrix} x_{S_i}[k] \\ x_{\bar{S}_i}[k] \end{bmatrix} \ . \tag{11}$$

The outputs seen by node i over the first L_i+1 time-steps of the linear iteration are given by

$$\begin{bmatrix} y_i[0] \\ y_i[1] \\ \vdots \\ y_i[L_i] \end{bmatrix} = \begin{bmatrix} E_{i,S_i} & 0 \\ E_{i,S_i}W_{i,S_i} & 0 \\ \vdots & \vdots \\ E_{i,S_i}W_{i,S_i}^{L_i} & 0 \end{bmatrix} \begin{bmatrix} x_{S_i}[0] \\ x_{\bar{S}_i}[0] \end{bmatrix}$$
$$= \begin{bmatrix} E_{i,S_i} \\ E_{i,S_i}W_{i,S_i} \\ \vdots \\ E_{i,S_i}W_{i,S_i}^{L_i} \end{bmatrix} x_{S_i}[0] \ . \qquad (12)$$

This shows that node i receives no information about any node in the set $\bar{S}_i$, regardless of the choice of weight matrix, and therefore cannot calculate any function of the node values from that set. Furthermore, the matrix multiplying $x_{S_i}[0]$ is the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}). Thus, maximizing the rank of the observability matrix for node i is equivalent to maximizing the rank of the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}).

To choose a set of weights that accomplishes this, we will use techniques from control theory pertaining to *linear structured systems* [18], [19], [20], [21]. Specifically, a linear system of the form

$$x[k+1] = Ax[k] \ ,$$
$$y[k] = Cx[k] \ ,$$

with state vector $x \in \mathbb{R}^N$ and output $y \in \mathbb{R}^p$ is said to be structured if each entry of the matrices A and C is either a fixed zero or an independent free parameter. A structured pair (A, C) is said to be *structurally observable* if the corresponding observability matrix has full column rank for at least one choice of free parameters. To analyze structural properties (such as observability) of these systems, one first associates a graph $\mathcal{H}$ with the structured pair (A, C) as follows. The vertex set of $\mathcal{H}$ is given by $\mathcal{X} \cup \mathcal{Y}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is the set of state vertices and $\mathcal{Y} = \{y_1, y_2, \ldots, y_p\}$ is the set of output vertices. The edge set of $\mathcal{H}$ is given by $\mathcal{E}_x \cup \mathcal{E}_y$, where $\mathcal{E}_x = \{(x_j, x_i) | A_{ij} \neq 0\}$ is the set of edges corresponding to interconnections between the state vertices, and $\mathcal{E}_y = \{(x_j, y_i) | C_{ij} \neq 0\}$ is the set of edges corresponding to connections between the state vertices and the output vertices. The following theorem characterizes the conditions on $\mathcal{H}$ for the system to be structurally observable. The terminology *$\mathcal{Y}$-topped path* is used to denote a path with end vertex in $\mathcal{Y}$.

Theorem 4 ([21]): The structured pair (A, C) is structurally observable if and only if the graph $\mathcal{H}$ satisfies both of the following properties:

1) Every state vertex $x_i \in \mathcal{X}$ is the start vertex of some $\mathcal{Y}$-topped path in $\mathcal{H}$.
2) $\mathcal{H}$ contains a subgraph that covers all state vertices, and that is a disjoint union of cycles and $\mathcal{Y}$-topped paths.

□

An appealing feature of structural properties is that they hold generically. In other words, if the structured system is observable for some particular choice of free parameters, it will be observable for almost any choice of parameters (i.e., the set of parameters for which the system is not observable has Lebesgue measure zero) [20], [21].

To apply the above results to the problem of maximizing the rank of the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}), we note that matrices W_{i,S_i} and E_{i,S_i} are essentially structured matrices, with the exception that the nonzero entries in E_{i,S_i} are taken to be 1 rather than free parameters. However, this is easily remedied by redefining the output available to node i in (11) as

$$\bar{y}_i[k] = \underbrace{\begin{bmatrix} \lambda_{i,1} & 0 & \cdots & 0 \\ 0 & \lambda_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{i,\deg_i+1} \end{bmatrix}}_{\Lambda_i} y_i[k]$$
$$= \Lambda_i \begin{bmatrix} E_{i,S_i} & 0 \end{bmatrix} x[k] \ ,$$

where the $\lambda_{i,j}$'s are a set of free parameters. The matrix $\Lambda_i E_{i,S_i}$ has a single independent free parameter in each row, and therefore qualifies as a valid structured matrix. The observability matrix for the structured pair $(W_{i,S_i}, \Lambda_i E_{i,S_i})$ is

$$\begin{bmatrix} \Lambda_i E_{i,S_i} \\ \Lambda_i E_{i,S_i} W_{i,S_i} \\ \vdots \\ \Lambda_i E_{i,S_i} W_{i,S_i}^{L_i-1} \end{bmatrix} = \begin{bmatrix} \Lambda_i & 0 & \cdots & 0 \\ 0 & \Lambda_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Lambda_i \end{bmatrix} \begin{bmatrix} E_{i,S_i} \\ E_{i,S_i} W_{i,S_i} \\ \vdots \\ E_{i,S_i} W_{i,S_i}^{L_i-1} \end{bmatrix} ,$$

and if the matrix Λ_i is invertible, the observability matrix for the pair $(W_{i,S_i}, \Lambda_i E_{i,S_i})$ will have the same rank as the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}). Thus, for the purposes of analyzing structural observability, we can assume without loss of generality that the nonzero entries in E_{i,S_i} are all ones (i.e., take Λ_i to be the identity matrix).

We can now use the above results on structured systems to prove the key result in Theorem 2 (introduced at the end of Section II).

Proof: [Theorem 2] From (12), we see that the output available to node i after $L_i + 1$ time-steps is simply the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}) multiplied by the vector $x_{S_i}[0]$. To prove the theorem, we only have to show that the pair (W_{i,S_i}, E_{i,S_i}) is structurally observable, and to accomplish this, we examine the associated graph $\mathcal{H}$. In this case, $\mathcal{H}$ is obtained as follows:

1) Take the subgraph of $\mathcal{G}$ induced by the nodes in S_i.
2) In this subgraph, add self-loops to every node to correspond to the free parameters w_{jj} on the diagonal of the matrix W_{i,S_i}.
3) Add $\deg_i + 1$ output vertices (denoted by the set $\mathcal{Y}_i$), and place a single edge from node i and each of its neighbors to vertices in $\mathcal{Y}_i$, corresponding to the single nonzero entry in each row of the matrix E_{i,S_i}.

Every node in $\mathcal{H}$ has a path to node i (by definition of the set S_i), and therefore to an output vertex in $\mathcal{Y}_i$ (e.g., the output vertex that has an edge from node i). This satisfies the first condition of Theorem 4. Furthermore, the self-loops on the nodes form a set of disjoint cycles that covers every node in $\mathcal{H}$. This satisfies the second condition in Theorem 4, and so the

pair (W_{i,S_i}, E_{i,S_i}) is structurally observable. This implies that the observability matrix multiplying $x_{S_i}[0]$ in (12) will have full column rank for almost any choice of W_{i,S_i} (and hence for almost any choice of W) for sufficiently large L_i. When W is a square matrix with N columns, the bound in (7) indicates that it will take at most $N - \deg_i$ time-steps for the observability matrix for the pair (W, E_i) to achieve its maximum rank. Since the matrix W_{i,S_i} has $|S_i|$ columns, we conclude that the observability matrix for the pair (W_{i,S_i}, E_{i,S_i}) will take at most $|S_i| - \deg_i$ time-steps to become full column rank. Node i can therefore recover the vector $x_{S_i}[0]$ from the outputs of the system over at most $|S_i| - \deg_i$ time-steps. ■

Theorem 2 indicates that node i can calculate any function of the initial values of nodes that have a path to node i, since it can reconstruct those initial values after running the linear iteration for a finite number of time-steps. Note that this is the most general result that can be obtained for a given (time-invariant) network, since if a node j does not have a path to node i in the network, it will be impossible for node i to calculate any function of node j's value (regardless of the protocol). Since the above theorem holds for any node i, and for all but a set of weights of measure zero, we immediately obtain the following corollary.

Corollary 1: Define the set $S = \cap_{i=1}^{N} S_i$, where

$$S_i = \{x_j | \text{ There exists a path from } x_j \text{ to } x_i \text{ in } \mathcal{G}\} \cup x_i \ .$$

In other words, S is the set of nodes that have a path to all nodes in the system. Then, for almost any choice of weight matrix W, the nodes can reach consensus on any function of the initial values of nodes in S after running the linear iteration for at most $\max_i(|S_i| - \deg_i)$ time-steps.

Corollary 1 indicates that the nodes can reach consensus after a finite number of iterations as long as the network has a spanning tree (i.e., there is at least one node that has a path to every other node). If the network is strongly connected (i.e., there is a path from every node to every other node), the nodes can reach consensus on any arbitrary function of the initial values. This result is more general than those currently existing in the literature (for time-invariant networks), which typically focus on the nodes reaching *asymptotic* consensus on a *linear functional* of the initial values (e.g., see [4], [3], [8], and the discussion in Sections II and IV-A).

V. Decentralized Calculation of the Observability Matrices

In the previous sections, we saw that if the weight matrix is chosen appropriately, the row space of the observability matrix for each node will contain enough information for that node to calculate the desired function of the initial values. Specifically, there will exist a matrix Γ_i satisfying (5) for each node i, and node i can use this matrix to calculate $Qx[0]$ from (6). If required, it can then calculate more general (nonlinear) functions of $Qx[0]$. However, finding Γ_i requires knowledge of the observability matrix $\mathcal{O}_{i,\nu_i-1}$. If the global topology of the graph and all of the weights are known *a priori*, then Γ_i can be calculated from the matrix $\mathcal{O}_{i,\nu_i-1}$ and conveyed to node i. Note that in graphs with time-invariant topologies, Γ_i only has to be computed once for each i. However, in practice, it may be the case that there is no opportunity to calculate the Γ_i's *a priori*, and therefore it will be necessary for the nodes to calculate these matrices using only local information. In this section, we show how each node i can calculate Γ_i in a decentralized manner. To accomplish this, we will assume that the nodes know N (or an upper bound for N). We will also assume that each node has a unique identifier, and that the nodes know their position in an appropriate ordering of the identifiers (e.g., the node with the j–th smallest identifier takes its position to be j). We assume without loss of generality that the vector $x[k]$ is consistent with this ordering (i.e., $x_i[k]$ is the value of the node whose position is i–th in the ordering).

As noted earlier, there are various ways for the nodes to choose their weights so that the row spaces of the observability matrices contain certain vectors. For example, if each node independently chooses a random set of weights for its update equation, the observability matrix will almost surely have maximum possible rank. Alternatively, if the nodes are only required to calculate the average of the initial values, they can choose the weights in (10), (9) or any other similar set of weights from the asymptotic consensus literature.

Once an appropriate set of weights is chosen, suppose the nodes perform N runs of the linear iteration, each for $N-1$ time-steps. For the j–th run, node j sets its initial condition to be 1, and all other nodes set their initial conditions to be zero. In other words, if $x_{*,j}[k]$ denotes the vector of node values during the k–th time-step of the j–th run, the nodes calculate $x_{*,j}[k+1] = Wx_{*,j}[k]$, $0 \leq k \leq N-2$, $1 \leq j \leq N$, where $x_{*,j}[0] = \mathbf{e}_j$. Suppose that for each of the N runs, node i stores the values it sees over the first $N - \deg_i$ time-steps. Each node i then has access to the matrix

$$\Psi_{i,L} = \begin{bmatrix} y_{i,1}[0] & y_{i,2}[0] & \cdots & y_{i,N}[0] \\ y_{i,1}[1] & y_{i,2}[1] & \cdots & y_{i,N}[1] \\ \vdots & \vdots & \ddots & \vdots \\ y_{i,1}[L] & y_{i,2}[L] & \cdots & y_{i,N}[L] \end{bmatrix} \ , \qquad (13)$$

for any $0 \leq L \leq N - \deg_i - 1$, where $y_{i,j}[k] = E_i x_{*,j}[k]$. Using (4), the above matrix can be written as

$$\Psi_{i,L} = \mathcal{O}_{i,L} \begin{bmatrix} x_{*,1}[0] & x_{*,2}[0] & \cdots & x_{*,N}[0] \end{bmatrix} \ ,$$

and since $x_{*,j}[0] = \mathbf{e}_j$, we see that $\Psi_{i,L} = \mathcal{O}_{i,L}$. Node i now has access to its observability matrix, and can find the matrix Γ_i and the smallest integer L_i such that $\Gamma_i \mathcal{O}_{i,L_i} = Q$ (for the desired matrix Q). For future initial conditions $x[0]$, node i can perform function calculation in the network by running the linear iteration (1) for $L_i + 1$ time-steps to obtain the values $y_i[0], y_i[1], \ldots, y_i[L_i]$. It can then immediately obtain $Qx[0]$ from (6), and use this to calculate the value $f(x_1[0], x_2[0], \ldots, x_N[0])$ for any function $f : \mathbb{R}^N \to \mathbb{R}^m$ (note that f can be completely arbitrary if $Q = I_N$).

It is important to note that the above discovery algorithm only needs to be run once (in time-invariant networks) in order for the nodes to obtain the Γ_i matrices. After they have obtained these matrices, they can use them to perform function calculation for arbitrary sets of initial conditions. In other words, the cost of discovering the Γ_i matrices will be

amortized over the number of times they are used by the nodes to perform function calculation. A full description of the algorithm (including the initialization phase) can be found in Fig. 1 and Fig. 2.

Remark 3: If the nodes only know an upper bound for N, the observability matrix obtained by each node will have one or more columns that are entirely zero. This is because there will be some runs of the linear iteration where all nodes set their initial values to be zero, under the assumption that some (nonexistent) node will be setting its value to 1. In such cases, the nodes can simply drop these zero columns from their observability matrices, and continue as normal. □

Remark 4: Note that the protocol described above for finding the observability matrix requires each node i to store $(\deg_i+1) \times N \times (N-\deg_i)$ values, because each output vector $y_{i,j}[k]$ has $\deg_i+1$ components, there are $N-\deg_i$ outputs stored per run, and there are N runs in total. However, it may be the case that the observability index ν_i for node i is smaller than $N-\deg_i$, and so node i may be storing more values than required (since after time-step ν_i, the rows of the observability matrix are linear combinations of the previous rows). This problem can be easily circumvented by having the nodes run the N linear iterations in *parallel*, as opposed to serially. In this way, the nodes construct the (observability) matrix $\Psi_{i,L}$ in (13) row-by-row, as opposed to column-by-column. If node i finds that the new rows of the observability matrix do not increase its rank, or if the existing rows of the observability matrix already contain the desired matrix Q, node i can stop storing values. Note, however, that all nodes still have to complete $N-1$ time-steps of each run in order to ensure that every node has an opportunity to find its observability matrix. This slight modification of the protocol would require each node to store at most $(\deg_i+1) \times N \times \nu_i$ values.

The communication cost incurred by the above protocol for discovering the observability matrix can be characterized as follows. Since each node transmits a single value on each outgoing edge at each time-step, and since there are $N-1$ time-steps per run, with N runs in total, each node i will have to transmit $N(N-1)$out-$\deg_i$ messages in order to run this protocol. Summing over all nodes in the network, there will be $\sum_{i=1}^{N} N(N-1)$out-$\deg_i = N(N-1)|\mathcal{E}|$ messages transmitted in total (since $\sum_{i=1}^{N}$ out-$\deg_i$ is equal to the number of edges in the graph [15]). Note that in wireless networks, a single transmission by node i will be equivalent to communicating a value along each outgoing edge of node i (since all neighbors of node i will receive the message after just one transmission), and thus each node would only have to transmit $N(N-1)$ messages in order to run the protocol, for a total of $N^2(N-1)$ messages in the network. □

Remark 5: It may be the case that the number of time-steps required by each node to calculate its desired function will vary from node to node (i.e., the value of L_i might be different for different i's). There are several different ways to handle this. One option is to have the nodes run the linear iteration for a full $N-1$ time-steps, even though they can calculate their desired function after L_i+1 time-steps (similar to what was suggested in Remark 4). However, this could cause the linear iteration to run for more time-steps than necessary (i.e., for $N-1$ time-steps, instead of $\max_i L_i+1$ time-steps). To get around this, one can have the nodes calculate the maximum of all the L_i's after they have calculated their Γ_i matrices. This can be done via a simple protocol where each node starts by broadcasting its own value of L_i, and then subsequently maintains and broadcasts only the largest value it receives from a neighbor. After at most N time-steps, all nodes will have the maximum value of L_i [5], and then the nodes can run subsequent linear iterations for $\max_i L_i+1$ time-steps.

If the nodes are required to reach consensus, another option to handle nonhomogeneous L_i's would be as follows. Each node i runs the linear iteration for L_i+1 time-steps, or until it receives the consensus value from a neighbor. If L_i+1 time-steps pass without receiving the consensus value, node i can calculate the value and broadcast it its neighbors, along with a flag indicating that it is the consensus value (and not just the next step in the linear iteration). In this way, slower neighbors of node i will receive the function value at most one time-step after node i calculates it. □

VI. Example

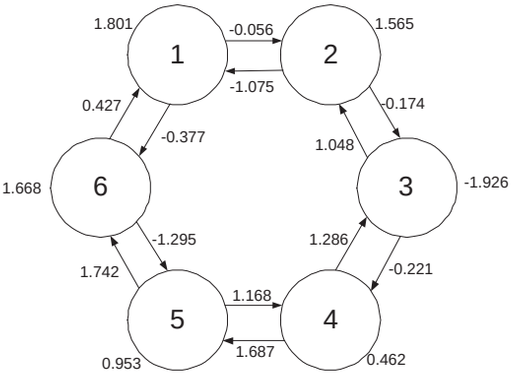

Fig. 3. Network with edge weights and self weights independently chosen from a uniform distribution on the interval $[-2, 2]$.

Consider the undirected network in Fig. 3. The objective in this network is for the nodes to reach consensus on the function

$$f(x_1[0], x_2[0], x_3[0], x_4[0], x_5[0], x_6[0]) = x_1^2[0] + x_2^2[0] + x_3^2[0] + x_4^2[0] + x_5^2[0] + x_6^2[0] \ . \quad (14)$$

The nodes do not know the entire topology, and there is no centralized decision maker to calculate the Γ_i matrices for each node. However, suppose that all nodes know that there are $N=6$ nodes in the system.

The first step is for the nodes to choose their weights. Since the graph is undirected (and hence strongly connected), Theorem 2 indicates that almost any choice of weights will allow the nodes to reconstruct the values $x_j[0]$, $1 \le j \le 6$ from the linear iteration (i.e., the matrix $Q = I_6$ will be in the row space of every observability matrix). The nodes can then use these values to calculate the function (14). For this example, we will have the nodes choose each of their weights

Initialization Phase: Calculation of Network Parameters

INPUT: Network $\mathcal{G}$, with N nodes $x_1, x_2, \ldots, x_N$, each with an ID and an associated function $f_i : \mathbb{R}^N \to \mathbb{R}^m$.

Each node i independently chooses a set of weights w_{ij} for the nodes in its neighborhood, and a weight w_{ii} for itself.
for $j = 1$ **to** N
  $x_j[0] = 1$, $x_i[0] = 0$ for all $i \neq j$.
  for $k = 0$ **to** N-2
    Each node i updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{l \in \mathcal{N}_i} w_{il} x_l[k].$$

    Each node i stores $y_{i,j}[k] = E_i x[k]$.
  end for
end for
Each node i forms the matrix Ψ_{i,L_i} in (13) and finds a value L_i and a matrix Γ_i satisfying $\Gamma_i \Psi_{i,L_i} = Q_i$, for an appropriate matrix Q_i which would allow node i to calculate $f_i(x_1[0], x_2[0], \ldots, x_N[0])$.
All nodes use a simple distributed protocol to determine $\max_{1 \leq i \leq N} L_i$.

OUTPUT: Γ_i, L_i and weights w_{ij} for each node i, and $\max_{1 \leq j \leq N} L_j$.

Fig. 1. The initialization phase of the protocol. This phase is used by the nodes to distributively determine the necessary information about the network in order to later perform distributed function calculation.

Calculating Functions of Arbitrary Initial Conditions

INPUT: Network $\mathcal{G}$, with N nodes $x_1, x_2, \ldots, x_N$, each with an (arbitrary) initial value $x_i[0]$ and an associated function $f_i : \mathbb{R}^N \to \mathbb{R}^m$. Each node i knows Γ_i, L_i, weights w_{ij} and $\max_{1 \leq j \leq N} L_j$.

for $k = 0$ **to** $\max_{1 \leq j \leq N} L_j$
  Each node i updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{l \in \mathcal{N}_i} w_{il} x_l[k].$$

  Each node i stores $y_i[k] = E_i x[k]$.
  if $k == L_i$ for some i **then**
    Node i calculates

$$\Gamma_i \begin{bmatrix} y_i[0] \\ y_i[1] \\ \vdots \\ y_i[L_i] \end{bmatrix} = Q_i x[0] \ ,$$

    which it uses to calculate $f_i(x_1[0], x_2[0], \ldots, x_N[0])$.
  end if
end for

OUTPUT: $f_i(x_1[0], x_2[0], \ldots, x_N[0])$ for each node i.

Fig. 2. The protocol to perform distributed function calculation via linear iterations. The inputs to this protocol can be obtained by running the initialization phase (given in Fig. 1), or can be calculated by a centralized entity and provided to each node i.

as an independent random variable uniformly distributed in the interval $[-2, 2]$. These weights are shown in Fig. 3.

Next, the nodes use the protocol described in Section V to discover their observability matrices. Specifically, they run $N = 6$ different linear iterations, with initial condition $x_{*,j}[0] = \mathbf{e}_j$ for the j–th linear iteration. As discussed in Remark 4, we will have the nodes run the N different linear iterations in parallel, so that they compute the observability matrix row-by-row. For instance, consider node 1 in Fig. 3. The matrix E_1 in (3) is given by

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \ .$$

After running the $N = 6$ different linear iterations for 3 time-

steps each, node 1 has access to the matrix

$$\begin{bmatrix} y_{1,1}[0] & y_{1,2}[0] & y_{1,3}[0] & y_{1,4}[0] & y_{1,5}[0] & y_{1,6}[0] \\ y_{1,1}[1] & y_{1,2}[1] & y_{1,3}[1] & y_{1,4}[1] & y_{1,5}[1] & y_{1,6}[1] \\ y_{1,1}[2] & y_{1,2}[2] & y_{1,3}[2] & y_{1,4}[2] & y_{1,5}[2] & y_{1,6}[2] \end{bmatrix} = \left[\begin{array}{rrrrrr} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1.80 & -1.08 & 0 & 0 & 0 & 0.43 \\ -0.06 & 1.56 & 1.05 & 0 & 0 & 0 \\ -0.38 & 0 & 0 & 0 & 1.74 & 1.67 \\ \hline 3.14 & -3.62 & -1.13 & 0 & 0.74 & 1.48 \\ -0.19 & 2.33 & -0.38 & 1.35 & 0 & -0.02 \\ -1.31 & 0.41 & 0 & 2.94 & 4.57 & 0.37 \end{array}\right],$$

where $y_{1,j}[k]$ is the output seen by node 1 at time-step k of the j–th run. At this point, node 1 finds that this matrix is of full column rank, and so it does not need to store any further values. However, it continues to perform the linear iterations for 2 more time-steps (for a total of $N-1=5$ time-steps), so that the observability matrix for every node in the system is guaranteed to be of full column rank.

After all N runs are complete, each node i has access to its observability matrix, and can calculate the matrix Γ_i satisfying $\Gamma_i \mathcal{O}_{i,L_i} = I_6$. We omit the values of the Γ_i matrices in the interest of space. In this example, it is found that $L_i = 2$ for all i, and so all nodes can reconstruct all initial values after $L_i + 1 = 3$ time-steps. At this point, the nodes find $\max_i L_i = 2$ via the protocol discussed in Remark 5, and so they all agree to stop running subsequent linear iterations after $\max_i L_i + 1 = 3$ time-steps.

Now that each node i has access to Γ_i, it can use this matrix to reconstruct any arbitrary set of initial values. Suppose that on the next run, the initial values on the nodes are given by $x[0] = \begin{bmatrix} -2 & 1 & 3 & -1 & 0 & 2 \end{bmatrix}'$. In order to calculate the function (14), the nodes run the linear iteration for three time-steps. The values of the nodes over those time-steps are given by

$$\begin{bmatrix} x[0] & x[1] & x[2] \end{bmatrix} = \begin{bmatrix} -2 & -3.823 & -10.3214 \\ 1 & 4.821 & 0.1735 \\ 3 & -7.238 & 11.6548 \\ -1 & -1.125 & -3.9157 \\ 0 & -4.277 & -11.2704 \\ 2 & 4.090 & 0.8129 \end{bmatrix}.$$

Using the outputs of the system $y_1[0] = E_1 x[0], y_1[1] = E_1 x[1]$ and $y_1[2] = E_1 x[2]$, node 1 can now reconstruct the initial values as

$$\Gamma_1 \begin{bmatrix} y_1[0] \\ y_1[1] \\ y_1[2] \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 3 \\ -1 \\ 0 \\ 2 \end{bmatrix}.$$

It then substitutes these values into (14) to obtain a value of 19. All other nodes calculate the function in the same manner, and the system reaches consensus in three time-steps.

Recall from Remark 2 that a lower bound on the number of time-steps required by each node to calculate the function (14) is given by the eccentricity of that node. Since both the radius and the diameter of the graph in Fig. 3 are equal to 3, the eccentricity of every node is also equal to 3 (i.e., for every node i, there is a node j at distance 3 from i), and the linear iteration achieves the lower bound in this example. In particular, no algorithm can achieve consensus in fewer than three time-steps (assuming it takes one time-step for a message to propagate along an edge), and so the linear iterative strategy is time-optimal for this network. In contrast, suppose we utilize a naive approach where all nodes send their values to a single node, which then calculates the desired function and broadcasts it back to all nodes. In the above network, this approach would require at least six time-steps (three time-steps for the central node to receive all values, and another three time-steps for the result to propagate back to all nodes).

VII. Summary and Future Work

As we have seen from our development, in any given fixed network, the nodes can calculate any function of the node values by following a linear iterative strategy with almost any choice of weights for a finite number of time-steps. We provided a specific class of weights that can be used to reach consensus on a linear functional of the values, and demonstrated that any weight matrix that provides asymptotic consensus (in time-invariant networks) also fits within this class. Finally, we showed that it is not necessary for the entire network topology to be known *a priori* in order to use our scheme, but that it is possible for the nodes to obtain the necessary information about the network by running the linear iteration with several different initial conditions.

There are a number of interesting directions for future research, including finding ways to choose the weight matrix to minimize the number of iterations required to calculate the desired functions, and incorporating robustness to errors in the nodes and links (including an extension to time-varying networks). We are actively pursuing these and other ideas in our research.

Appendix
Proof of Theorem 3

Proof: Let R_i be a matrix whose rows form a basis for the row space of $\mathcal{O}_{i,\nu_i-1}$. Thus, R_i represents all linear functionals of $x[0]$ that can be obtained from the output of the system (i.e., it completely characterizes all linear functionals that are calculable by node i). Define the matrix

$$P_i = \begin{bmatrix} R_i \\ T_i \end{bmatrix}, \tag{15}$$

where T_i is chosen to make P_i square and invertible. If we use P_i to perform a similarity transformation on the system (3) with output $y_i[k] = E_i x[k]$, we obtain the *observable canonical form*

$$\bar{W} \equiv P_i W P_i^{-1} = \begin{bmatrix} W_{i,o} & 0 \\ W_{i,o\bar{o}} & W_{i,\bar{o}} \end{bmatrix}, \tag{16}$$
$$\bar{E}_i \equiv E_i P_i^{-1} = \begin{bmatrix} E_{i,o} & 0 \end{bmatrix}.$$

In this form, the pair $(W_{i,o}, E_{i,o})$ is observable, and in particular, all observable eigenvalues of the pair (W, E_i) are contained in the submatrix $W_{i,o}$ [14], [16].

Let $\bar{C}_{i,o}$ be the matrix whose rows contain all left eigenvectors of $W_{i,o}$ corresponding to the observable eigenvalues of the pair (W, E_i). Thus, we have $\bar{C}_{i,o}W_{i,o} = J_{i,o}\bar{C}_{i,o}$, where $J_{i,o}$ is the set of all Jordan blocks corresponding to observable eigenvalues of the pair (W, E_i) [13], [14]. Now note that

$$\begin{bmatrix}\bar{C}_{i,o} & 0\end{bmatrix}\begin{bmatrix}W_{i,o} & 0\\ W_{i,o\bar{o}} & W_{i,\bar{o}}\end{bmatrix} = J_{i,o}\begin{bmatrix}\bar{C}_{i,o} & 0\end{bmatrix} \ .$$

From the definition of $\bar{W}$ in (16), we have

$$\begin{bmatrix}\bar{C}_{i,o} & 0\end{bmatrix}P_i W = J_{i,o}\begin{bmatrix}\bar{C}_{i,o} & 0\end{bmatrix}P_i \ ,$$

and using (15), we get $\bar{C}_{i,o}R_i W = J_{i,o}\bar{C}_{i,o}R_i$. This implies that the rows of the matrix $\bar{C}_{i,o}R_i$ are the left eigenvectors of W corresponding to observable eigenvalues of the pair (W, E_i). Since the rows of R_i form a basis for the row space of $\mathcal{O}_{i,\nu_i-1}$, we see that all left eigenvectors of the matrix W corresponding to observable eigenvalues of the pair (W, E_i) can be obtained as a linear combination of rows in $\mathcal{O}_{i,\nu_i-1}$, thereby completing the proof. ■

Remark 6: Note that the decomposition of the weight matrix W in (11) has the same structure as the observable canonical form (16). The difference between the two forms is that the observable canonical form is obtained from a similarity transformation of a particular *numerical* weight matrix, and furthermore, the similarity transformation matrix P_i does not have to be a permutation matrix. In contrast the decomposition in (11) is obtained by simply permuting the rows and columns of a *structured* weight matrix W. For a given network, the size of the matrix $W_{i,o}$ in (16) will be no larger than the size of the matrix W_{i,S_i} in (11), and can be smaller for certain choices of the weight matrix. □

REFERENCES

[1] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[2] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004, pp. 20–27.

[3] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the American Control Conference*, 2005, pp. 1859–1864.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[5] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[6] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[7] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, 2005, pp. 2996–3000.

[8] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[10] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[11] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of the American Control Conference*, 2006, pp. 3551–3556.

[12] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, Nov. 2006.

[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[14] C.-T. Chen, *Linear System Theory and Design*. Holt, Rinehart and Winston, 1984.

[15] D. B. West, *Introduction to Graph Theory*. Prentice-Hall Inc., Upper Saddle River, New Jersey, 2001.

[16] D. S. Bernstein, *Matrix Mathematics*. Princeton University Press, 2005.

[17] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 63–70.

[18] C.-T. Lin, "Structural controllability," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 201–208, June 1974.

[19] S. Hosoe, "Determination of generic dimensions of controllable subspaces and its application," *IEEE Transactions on Automatic Control*, vol. 25, no. 6, pp. 1192–1196, Dec. 1980.

[20] K. J. Reinschke, *Multivariable Control A Graph-Theoretic Approach*. Springer-Verlag, 1988.

[21] J.-M. Dion, C. Commault, and J. van der Woude, "Generic properties and control of linear structured systems: a survey," *Automatica*, vol. 39, no. 7, pp. 1125–1144, July 2003.